Available online at www.sciencedirect.com

ScienceDirect

International Journal of Approximate Reasoning
44 (2007) 32–44

INTERNATIONAL JOURNAL OF APPROXIMATE REASONING

www.elsevier.com/locate/ijar

Improving the accuracy while preserving the interpretability of fuzzy function approximators by means of multi-objective evolutionary algorithms ☆

Jesús González *, Ignacio Rojas, Héctor Pomares, Luis J. Herrera, Alberto Guillén, José M. Palomares [1], Fernando Rojas

Department of Computer Architecture and Computer Technology,
E.T.S. Ingenierías Informática y de Telecomunicacíon, University of Granada, Granada E-18071, Spain

Received 18 July 2005; received in revised form 20 January 2006; accepted 6 February 2006
Available online 21 July 2006

Abstract

The identification of a model is one of the key issues in the field of fuzzy system modeling and function approximation theory. An important characteristic that distinguishes fuzzy systems from other techniques in this area is their transparency and interpretability. Especially in the construction of a fuzzy system from a set of given training examples, little attention has been paid to the analysis of the trade-off between complexity and accuracy maintaining the interpretability of the final fuzzy system. In this paper a multi-objective evolutionary approach is proposed to determine a Pareto-optimum set of fuzzy systems with different compromises between their accuracy and complexity. In particular, two fundamental and competing objectives concerning fuzzy system modeling are addressed: fuzzy rule parameter optimization and the identification of system structure (i.e. the number of membership functions and fuzzy rules), taking always in mind the transparency of the obtained system. Another key aspect of the algorithm presented in this work is the use of some new expert evolutionary operators, specifically designed for the problem of fuzzy function

☆ Partially supported by the Spanish *Ministerio de Ciencia y Tecnología* under projects TIC2002–11352–E and TIN2004–01419.

* Corresponding author.
E-mail address: jgonzalez@atc.ugr.es (J. González).
[1] Present address: Department of Electrotechnics and Electronics, Escuela Politécnica Superior, University of Córdoba, E–14071 Córdoba, Spain.

0888-613X/$ - see front matter
doi:10.1016/j.ijar.2006.02.006

approximation, that try to avoid the generation of worse solutions in order to accelerate the convergence of the algorithm.

Keywords: Function approximation; Fuzzy systems; Multi-objective optimization; Evolutionary algorithms

1. Introduction

The problem of estimating an unknown function $\mathscr{F}$ from samples of the form $\{(\vec{x}^k, y^k); k = 1, 2, \ldots, n;$ with $y^k = \mathscr{F}(\vec{x}^k) \in \mathbb{R}$, and $\vec{x}^k \in \mathbb{R}^d\}$ (i.e. function approximation from a finite number of data points), is one of the key issues in the field of fuzzy system modeling and function approximation theory [6,8,11,22,46]. The principal goal is to learn an unknown functional mapping between input and output vectors, using a set of known training samples. Once this mapping is generated, it can be used for predicting the output values given new input vectors. Inputs and outputs can be continuous and/or categorical variables. This paper is concerned with continuous output variables, thus considering regression or function approximation problems, as opposed to classification problems in which the output variable is categorical [8].

Recently, model-free systems, such as artificial neural networks or fuzzy systems [8,22,36–38], have been proposed to avoid the knowledge-acquisition bottleneck. Fuzzy systems provide an attractive alternative to the "black boxes" characteristic of neural network models, because their behavior can be easily explained by a human being. In fact, the popularity and practicality of fuzzy systems derives from their ability to express relations that are either complex or not sufficiently understood, in terms of linguistic rules.

Although one of the most widely used approaches for the design of a fuzzy system is the use of a complete table of rules, in this work we have designed a fuzzy system architecture based on free rules [22] or fuzzy patches [11], since the former approach suffers the *curse of dimensionality* as the number of inputs and membership functions is increased, and the latter one lets to allocate fuzzy rules only in the areas of the input space that contribute to minimize the output error, thus producing systems that are better fitted, with fewer parameters and more interpretable. These free rules or fuzzy patches have been implemented using gaussian membership functions (see Fig. 1), because as they are continuous and differentiable, they produce a smoother output and improve the system's interpolation capability. Fixed a position $\vec{p}^i$ and a width w^i in the input space, the following equation is used to model a fuzzy patch

$$\alpha_i(\vec{x}) = \exp\left(-\sum_{j=1}^{d}\left(\frac{\left\|x_j - p_j^i\right\|}{w^i}\right)^2\right), \tag{1}$$

where d is the number inputs to the system, $\vec{x} = (x_1, \ldots, x_d)$ is an input vector, and $i = 1, \ldots, m$ is the number of patches in the system.

Finding an adequate configuration for a fixed-structure system is a complex problem with many local optima, due to the non linear dependencies between the parameters defining the model, and this problem becomes harder if we also have to adjust complexity of the

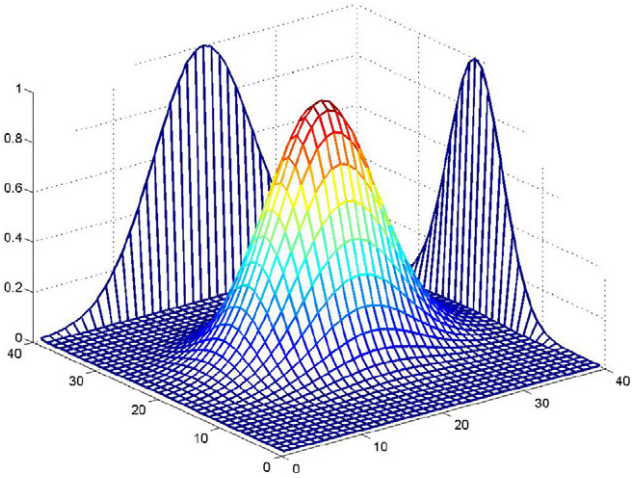

Fig. 1. Gaussian fuzzy patch.

model (number of patches) simultaneously, because different solutions can be found with different compromises between their accuracy and their generalization properties.

The selection of the adequate compromise between the accuracy and the complexity of the model may be subjective and usually depends on the problem to be solved, so we have chosen a Multi-Objective Evolutionary Algorithm (MOEA) to find a complete set of Pareto-optimal solutions. This set is presented to the final user, who can select the most appropriate model for his particular application.

Multi-Objective Optimization (MOO) techniques have been applied to design fuzzy systems since the mid-1990s. For example, in [14] a classical MOO technique is used to aggregate multiple objectives in a single function to optimize a fuzzy system. MOEAs have also been used to design fuzzy systems in the last years. In [19], a MOEA is used to select linguistic rules for pattern classification problems; in [15] a single step EA is proposed to find multiple Pareto-optimal solutions to the problems of generation and/or tuning of fuzzy models. Several multi-objective evolutionary approaches are proposed in [23–25] to consider different objectives dealing with transparence and compactness. It is also noteworthy the work proposed by Ishibuchi and Yamamoto [20], where a MOEA is used to construct an ensemble of fuzzy rule-based classifiers with high diversity, and the work proposed in [44], where a MOEA is used to extract interpretable rule-based knowledge from data.

MOEAs can also obtain better solutions if some expert knowledge about the problem to be solved is incorporated [3]. So, in this paper we propose the incorporation of this expert knowledge to the well known *Nondominated Sorting Genetic Algorithm* II (NSGA-II) [10]. This specialization is performed first, with an initialization of the population with relatively good solutions through the use of several heuristics, and second, by designing some problem-specific crossover and mutation operators for the problem of approximating a target function with a set of fuzzy patches.

The paper is arranged as follows. First, the training algorithm, with its expert initialization and evolutionary operators, is described in Section 2. Then, Section 3 presents some experimental results and comparisons with other approaches used to forecast the Mackey–Glass time series, and finally, in Section 4 some concluding remarks are pointed out.

2. The training algorithm

An EA is a probabilistic search algorithm that evolves a population of individuals until a convergence criterion is met. In this context, each individual represents a potential solution to the problem with a particular structure and configuration. This evolution is performed by the application of mutation and crossover operators in each generation and by the application of the "survival of the fittest" principle to keep the probably better solutions for the next generation. In multimodal and multi-objective problems it is also necessary to include a niche-formation method to distribute the population over the optimal solutions and avoid the premature convergence of the algorithm.

As EAs are generic optimization tools, there are many possibilities to fine-tune them for a specific problem. They are able to use different data structures to represent individuals, different genetic operators to transform them, different ways of initializing the population, additional methods to handle restrictions, different schemes for the selection process, different termination criteria, different niching and fitness sharing strategies, different ways of handling Multi-Objective Problems (MOPs), etc., so, in this section we will explain the configuration used for the problem of constructing fuzzy systems from training examples.

Basically, we have implemented a variation of NSGA-II, a fast and elitist MOEA able to find a good spread of solutions for a MOP that avoids the problem of fixing a sharing parameter to preserve the diversity of the population. In the reamining of this section we will explain the changes we have applied to NSGA-II to adapt it to the problem of function approximation from a set of traning data using a patch-based fuzzy system. The rest of the features of the algorithm not commented below have been kept as in the original NSGA-II. For more details about NSGA-II, readers are encouraged to refer to [10].

2.1. Initial population

Although ideally a MOEA can converge to the best solutions in an infinite number of iterations from any sufficiently diverse initial population, in practice we want to obtain a set of appropriate solutions in a reasonable computation time. Thus, if the initial individuals can be constructed using some heuristics or expert knowledge, we can save many generations of evolutionary search. With this idea in mind, and knowing that the consequents of the fuzzy model can be obtained optimally, we can apply some heuristics to fix the position $\vec{p}^i$ and width w^i of the patches that form each individual in the population.

The initialization procedure also uses three parameters, s, $m_{\min}$, and $m_{\max}$, representing the size of the population, and the minimum and maximum threshold we want for the number of patches in each initial fuzzy system. The two steps used to obtain the initial population are described below.

2.1.1. First step

To generate a set of initial configurations we have applied CFA [17] and other clustering techniques such as the *C*-means [12] and the *Enhanced LBG* (ELBG) algorithm [32]. These three techniques are applied to the set of training examples with all the possible number of patches m in the range $[m_{\min},m_{\max}]$, obtaining a set of $3 \cdot (m_{\max} - m_{\min} + 1)$ initial configurations for the positions.

Once we have obtained a variety of configurations for the number of fuzzy patches and their positions, it is time to obtain their widths. Thus, CIV and KNN with $k \in \{1,2,3\}$ are

applied to all the configurations for the positions obtained above, obtaining a set of $4 \cdot 3 \cdot (m_{\max} - m_{\min} + 1)$ possible initial fuzzy systems.

After obtaining the positions and widths of this initial set of fuzzy systems, we can calculate their optimum consequents using SVD.

2.1.2. Final step

After executing the initial step, we obtain a set of $4 \cdot 3 \cdot (m_{\max} - m_{\min} + 1)$ fuzzy systems. This number of potential solutions will probably be different from s, thus we will need to generate or eliminate some solutions.

If we need more individuals to form a set of s initial solutions, we can generate the remaining ones applying mutation operators (see Section 2.3) to the solutions obtained in the initial step. On the contrary, if we have to eliminate some individuals to reduce the population size to s, we perform a nondominating sorting of the population (see Section 2.2) and erase the worst solutions.

Generating the initial population in this way, along with the expert evolutionary operators described below, is expected to be sufficient to find very good solutions with relatively small populations and in a reasonable number of iterations.

2.2. Evaluation function

The evaluation function is the part of the EA that is responsible for guiding the search towards the global optima. For this purpose, it must be designed to be able to compare two different individuals in order to distinguish the better solutions from the others. We also have to take into account that the set of solutions is not completely ordered for MOPs. In our case, we face a problem with two competing objectives: the accuracy and the complexity of the model. The accuracy of each fuzzy approximator is estimated with the Normalized Mean Squared Error (NMRSE) index, calculated as

$$f_1 = \mathrm{N}RMSE = \sqrt{\frac{\sum_{k=1}^{n} (y^k - F(\vec{x}^k))^2}{\sum_{k=1}^{n} (y^k - \bar{y})^2}}, \tag{2}$$

where $F(\vec{x}^k)$ is the approximated output for $\vec{x}^k$ obtained with the fuzzy model, y^k is its target output, and $\bar{y}$ is the mean of the n training outputs. The other objective, the complexity of the model, is estimated with its number of fuzzy patches m, that is,

$$f_2 = m. \tag{3}$$

The Pareto-dominance criterion allows comparing two different solutions, but it can not measure the difference between them. There have been proposed several approaches in the literature to overcome this problem, such as the *Multi-Objective Genetic Algorithm* (MOGA) presented in [13], the *Nondominated Sorting Genetic Algorithm* (NSGA) described in [41] or the *Nondominated Sorting Genetic Algorithm-II* (NSGA-II) [10]. As the MOEA presented in this paper is based on NSGA-II, we also use the *crowded-comparison operator* ($\prec_n$) [10], which orders solutions according to their rank and when two solutions have the same rank, prefers the solution located in a lesser crowded region.

2.3. Reproduction process

Genetic operators are applied to each individual within the population, with an application probability of p_c for the crossover operators, and of p_m for the mutation operators. Due to the problems of establishing these probabilities a priori, the algorithm presented here implements a dynamic adaptation mechanism of p_c and p_m [40] which chooses the values that are appropriate at all times, based on the state of convergence of the population. Once the probabilities have been chosen, the genetic operators described below are applied.

The evolutionary operators described in this section have been specifically designed for the problem of optimizing the parameters of a fuzzy system. These new operators apply random changes to the individuals they affect to maintain the diversity in the population and to provide mechanisms to escape from local minima [16,18], but also try to avoid the application of changes that could worsen the fitness of the solutions.

We will present a crossover operator, together with some mutation operators that can be organized into two groups: operators to change the structure of the fuzzy system and operators to adjust its parameters. The former group contains *SVD-based Pruning* (SVDP), *OLS based-Pruning* (OLSP), and the *Splitting of Fuzzy Systems* (SPLIT), while the latter group is composed of the *OLS-based Mutation* (OLSM), and the *SVD-based Mutation* (SVDM).

2.3.1. Crossover of fuzzy systems

This operator takes two fuzzy systems and returns two offspring combining the genetic information from their ancestors. The descendants are generated by interchanging several fuzzy patches in the original solutions. Some patches are selected randomly in one of the ancestors and are replaced by the closest patches in the input space belonging to the second progenitor, in order to avoid the generation of two new fuzzy systems that could leave input regions uncovered. After the exchange of information, the optimum consequents are obtained for the descendants using the Cholesky method [35].

2.3.2. SVD-based pruning

Singular Value Decomposition has been widely applied to detect the less relevant rules in a fuzzy system [30,48,49]. This decomposition returns a set of singular values (each one associated with one patch of the model) with an important property: the most relevant patches are detected because they have the highest singular values.

Taking this information into account, this mutation operator assigns a pruning probability to each patch that is inversely proportional to its associated singular value. Less important rules are assigned a greater pruning probability than more relevant ones. Once these pruning probabilities have been calculated, a patch is randomly selected and deleted, and the optimum consequents for the remaining patches are obtained.

2.3.3. OLS-based pruning

Another orthogonal transformation used in the literature to detect less relevant rules is OLS [7,45,49]. This method also assigns a relevance value to each patch, but with an important difference: OLS takes into account the expected output for each input vector in the training set. Thus, the relevance of each patch is closely related to its contribution to the reduction of the training error. The patches making a bigger contribution to the

training error reduction will be more sensitive to the pruning than those making a smaller contribution. Thus, this mutation operator assigns a pruning probability to each patch inversely proportional to its error reduction ratio and selects one patch to be deleted according to this probability distribution. After the deletion the optimum consequents of the model are recalculated.

2.3.4. Splitting of patches

The objective of this mutation operator is to detect the input areas that are worse modeled by the fuzzy system, i.e. those with a higher approximation error, and to increase the number of patches in these areas in order to increase the variance of the data explained by the fuzzy system. To carry out this task, the operator estimates the contribution of each patch to the whole approximation error using the following expression

$$e^j = \sum_{k=1}^{n} \frac{\alpha_j(\vec{x}^k)}{\sum_{i=1}^{m} \alpha_i(\vec{x}^k)} |F(\vec{x}^k) - y^k|, \quad j = 1, \ldots, m. \tag{4}$$

A high value of e^j means that the j-th rule is not able to model correctly the training data that most activate it, so, it would be desirable to increment the number of fuzzy rules in this input zone, in order to minimize the approximation error caused by the training examples. Thus, the mutation operator assigns a splitting probability to each rule proportional to its contribution to the approximation error. This means that those rules with a higher contribution to the approximation error will have more probability of being split. Once that all the rules have been assigned a splitting probability, one of them is randomly selected according to these probabilities distribution.

After the j-th rule has been selected, the two-means algorithm is run with the input examples that are closer to it than to any other rule, obtaining two new positions for two new patches, which will substitute the j-th rule in the affected fuzzy system. The widths of the MFs defining the antecedents of these two new rules are calculated using the KNN heuristic [26,29], and the optimum consequents for all the rules of the new fuzzy system are obtained.

2.3.5. SVD-based mutation

Another way of hybridization between SVD and a mutation operator is to select the fuzzy rule to be altered uniformly (all the rules have the same likelihood of being chosen) and apply a random displacement according to its associated singular value. Rules with small singular values will undergo large changes, while only small perturbations will be applied to sensitive patches.

This behavior can be implemented by applying a random shift to the position or the width of the selected patch whose modulus varies inversely with the magnitude of its singular value.

2.3.6. OLS-based mutation

This evolutionary operator implements exactly the same steps as SVDM, but uses OLS instead of SVD to detect the relevance of the patches. All the patches have the same likelihood of being altered, and when one of them is chosen, its position or its width undergoes a random displacement that is inversely proportional to its error reduction ratio.

2.4. Termination criterion

As the MOEA is used to find a good Pareto-optimum frontier which later will be fine-tuned using a minimization algorithm, the solutions found by the evolutionary algorithm only have to be reasonably close to a good set of solutions in order that the minimization algorithm can reach them. There have been several studies to fix a threshold for the maximum number of generations for an evolutionary algorithm [1,2,28,31,42,43], but all of them are based on a binary representation of the solutions and for very simple problems; besides, they all fix a maximum number of generations, thus making the algorithm execute more iterations than necessary.

The proposed algorithm incorporates a termination criterion based on the linear regression of the fitness of the best individuals found in the last iterations to detect the convergence of the population. This criterion calculates the slope of the linear regression of the best individuals' fitness found in the last v iterations (one slope is calculated for each solution i in the Pareto-optimum frontier), i.e.,

$$\mathrm{slope}_i(v) = \frac{\sum_{k=t-v+1}^{t}(k-t)O_i^k - \frac{\left(\sum_{k=t-v+1}^{t} O_i^k\right)\left(\sum_{k=t-v+1}^{t} k-t\right)}{v}}{\sum_{k=t-v+1}^{t}(k-t)^2 - \frac{\left(\sum_{k=t-v+1}^{t} k-t\right)^2}{v}}, \tag{5}$$

where t indicates the actual generation and O_i^k is the fitness of the Pareto-optimum solution with i patches in the k-th generation. Once calculated the slopes for all the solutions in the Pareto-optimum frontier, the evolutionary algorithm will stop if

$$\max_i\{\mathrm{slope}_i\} < \mathrm{Stop}_{\mathrm{th}}, \tag{6}$$

where $\mathrm{Stop}_{\mathrm{th}}$ is the stopping threshold to detect the convergence of the population. As the computation time is also an important factor, this termination criterion also uses the parameter *maxGens* to stop the algorithm after a given maximum number of generations in case the population has not converged yet.

2.5. Fine-tuning the solutions

Evolutionary algorithms comprise a powerful search tool, but they are incapable of reaching a solution that precisely fits the problem in the time available; nevertheless, they do achieve an intermediate solution that could serve as a starting point for a local search process that would achieve a solution with the desired degree of precision. To speed up the evolution of the system, we introduce a gradient-descent step into each generation, which is applied only to the Pareto-optimum individuals in the population to minimize their approximation error. This step, which represents an improvement on the best solutions of the previous generation, should not be carried out in too many iterations, as gradient descent is a very costly process. It is intended to refine the best solutions generated in the current generation within the population, so that in the following generation they can be crossed with others and thus generate partially-fitted solutions; this local fit can then be transmitted to the whole population in successive generations.

When the evolutionary algorithm has finished searching, the Pareto-optimal solutions found are subjected to a gradient descent, this time with a higher number of iterations to reach the closest local optima to the Pareto-optimal region returned by the algorithm.

3. Experimental results

The MOEA proposed in this paper has also been tested with the time series generated by the Mackey–Glass time-delay differential equation [27]

$$\frac{ds(t)}{dt} = \alpha \cdot \frac{s(t-\tau)}{1+s^{10}(t-\tau)} - \beta s(t). \tag{7}$$

Following previous studies [47], the parameters were fixed to $\alpha = 0.2$, $\beta = 0.1$, thus obtaining a chaotic time series with no clearly defined period; it does not converge or diverge, and is very sensitive to initial conditions.

As in [21], the time series values at integer points were obtained applying the fourth-order Runge–Kutta method to find the numerical solution for the above equation. The values $s(0) = 1.2$, $\tau = 17$, and $s(t) = 0$ for $t < 0$ were assumed. This data set can be found in the file `mgdata.dat` belonging to the FUZZY LOGIC TOOLBOX OF MATLAB.[2]

Following the conventional settings to perform a long term prediction of these time series, we predict the value $s(t+85)$ from the current value $s(t)$ and the past values $s(t-6)$, $s(t-12)$, and $s(t-18)$; thus, the training vectors for the model have the following format

$$[s(t-18), s(t-12), s(t-6), s(t), s(t+85)]. \tag{8}$$

The first 500 input vectors were used to train the model and the next 500 vectors were used to test the fuzzy systems obtained. The proposed algorithm was run several times with a population of 90 individuals until the population has converged or a maximum number of generations $maxGens = 1000$ is reached. Table 1 compares the mean and standard deviation over five executions of the solutions found by the proposed algorithm with other presented in the literature.

Some of the approaches being compared are based on *Radial Basis Function Neural Networks* (RBFNNs), such as the *Resource Allocation Network* (RAN) algorithm [34], which iteratively constructs an RBFNN analyzing the novelty of the input data, or the modifications of RAN proposed in [39], which include the *Givens QR decomposition* (RAN–GQRD) to obtain the weights of the net and a *pruning criterion* (RAN–P–GQRD) to reduce the complexity of the net. The results are compared with other fuzzy systems too. In [4] two different algorithms to train fuzzy systems are presented, one using brute force and another incremental, and it is shown that the brute force approach presents an unstable behavior as the number of rules is increased and besides it does not reach the approximation errors obtained by the incremental algorithm. The last approach [9] applies *Breeder Genetic Algorithms* (BGAs) to train MLPs. Again, it should be noted that the proposed algorithm is able to find a set of Pareto-optimum solutions that dominate all the solutions in the table. Fig. 2 summarizes graphically the results.

For most applications, and with suitable guards to prevent overfitting, the model with minimum approximation error in the Pareto-optimum frontier would seem the best one.

[2] MATLAB is a trademark of *The Math Works Inc.*

Table 1
Comparison of the proposed algorithm with others applied in the literature to predict the $s(t+85)$ value of the Mackey–Glass time series; m represents the number of rules or RBFs (depending on the model), and n_p is the number of free parameters

Algorithm		m	n_p	Test NRMSE
MLP + BGA [9]		16	80	0.2666
RAN [34]	$\epsilon = 0.1$	57	342	0.378
	$\epsilon = 0.05$	92	552	0.376
	$\epsilon = 0.02$	113	678	0.373
	$\epsilon = 0.01$	123	738	0.374
RAN–GQRD [39]	$\epsilon = 0.1$	14	84	0.206
	$\epsilon = 0.05$	24	144	0.170
	$\epsilon = 0.02$	44	264	0.172
	$\epsilon = 0.01$	55	330	0.165
RAN–P–GQRD [39]	$\epsilon = 0.1$	14	84	0.206
	$\epsilon = 0.05$	24	144	0.174
	$\epsilon = 0.02$	31	186	0.160
	$\epsilon = 0.01$	38	228	0.183
Fuzzy systems [4]	Brute force	10	190	0.1086
		11	206	0.1098
		12	228	0.1026
		13	247	0.2235
		14	266	0.1568
		15	285	0.1028
	Incremental	14	266	0.0965
Proposed algorithm		13	78	0.1984 ± 0.0163
		14	84	0.1856 ± 0.0159
		15	90	0.1693 ± 0.0425
		16	96	0.1548 ± 0.0233
		17	102	0.1453 ± 0.0146
		18	108	0.1354 ± 0.0109
		19	114	0.1233 ± 0.0154
		20	120	0.1172 ± 0.0124
		21	126	0.1112 ± 0.0123
		22	132	0.1045 ± 0.0099
		23	138	0.1023 ± 0.0126
		24	144	0.0945 ± 0.0074
		25	150	0.0855 ± 0.0058

Nevertheless, in other applications, such as classification [33], or the control of a plant [5], the user may be interested in other Pareto-optimum solutions with fewer parameters (rules), which although less accurated, are more interpretable, in order to understand the functioning of the system that has been modeled with the fuzzy system.

4. Conclusions

This study presents a different approach to the study of fuzzy systems in which the rules are adapted to the problem by covering the zones of the input space that most contribute to reducing the global approximation error of the system.

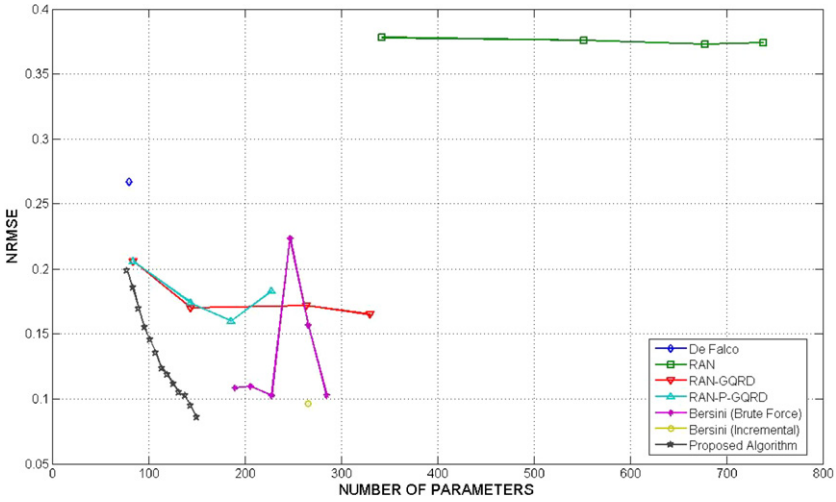

Fig. 2. Comparison of the proposed algorithm with others applied in the literature to predict the $s(t+85)$ value of the Mackey–Glass time series.

We have also presented a modification of the well known NSGA-II algorithm for the problem of modeling a fuzzy system for function approximation from a set of training data that applies expert mutation operators to avoid the generation of less fitted solutions, hybridizing the fundamental principles of genetic algorithms with those of classical optimization algorithms, thus achieving an algorithm that provides the power of evolutionary algorithms but at the same time one that fits the solutions with the desired degree of precision. The simulations performed show that the synergy of the different paradigms and techniques used produce excellent results for the construction of fuzzy systems.

Other important issue is the possibility of obtaining a wide range of solutions with different compromises between the accuracy and the complexity of the models in only one run of the algorithm, therefore the user can select the trade-off between accuracy and interpretability. It is also important to note that the proposed methodology is robust, being able to find similar set of solutions starting from different initial populations, as shown in Section 3.

References

[1] J.T. Alander, Genetic algorithms and other "natural" optimization methods to solve hard problems – a tutorial review. Technical Report 96-1, Department of Information Technology and Production Economics, University of Vaasa, Finland, 2000.

[2] T. Bäck, Generalized Convergence models for Tournament and (μ,λ)-Selection, in: L. Eshelman (Ed.), Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco, CA, 1995, pp. 2–8.

[3] T. Bäck, D.B. Fogel, Z. Michalewicz (Eds.), Handbook of Evolutionary Computation, Institute of Physics Publishing and Oxford University Press, Bristol, New York, 1997.

[4] H. Bersini, A. Duchateau, N. Bradshaw, Using incremental learning algorithms in the search for minimal and effective fuzzy models, in: Proceedings of the 6th IEEE International Conference on Fuzzy Systems, IEEE Computer Society Press, Barcelona, Spain, 1997, pp. 1417–1422, July.

[5] C.Y. Chen, B.D. Liu, Linguistic hedges and fuzzy rule based systems, in: J. Casillas, O. Cordón, F. Herrera, L. Magdalena (Eds.), Accuracy Improvements in Liguistic Fuzzy Modeling, Studies in Fuzziness and Soft Computing, vol. 129, Springer, 2003, pp. 165–192.

[6] D.S. Chen, R.C. Chain, A robust back propagation learning algorithm for function approximation, IEEE Transactions on Neural Networks 5 (3) (1994) 467–479.
[7] S. Chen, C.F.N. Cowan, P.M. Grant, Orthogonal least squares learning algorithm for radial basis functions, IEEE Transactions on Neural Networks 2 (1991) 302–309.
[8] V. Cherkassky, D. Gehring, F. Mulier, Comparison of adaptive methods for function estimation from samples, IEEE Transactions on Neural Networks 7 (4) (1996) 969–984.
[9] I. De Falco, A. Della Cioppa, A. Iazzetta, P. Natale, E. Tarantino, Optimizing neural networks for time series prediction, in: R. Roy, T. Furuhashi, P.K. Chawdhry (Eds.), Proceedings of the third On-line World Conference on Soft Computing (WSC3), Advances in Soft Computing – Engineering Design and Manufacturing, Internet, Springer Verlag, 1998.
[10] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA–II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.
[11] J.A. Dickerson, B. Kosko, Fuzzy function approximation with ellipsoidal rules, IEEE Transactions on Systems Man and Cybernetics – Part B 26 (4) (1996) 542–560.
[12] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, John Wiley, New York, 1973.
[13] C.M. Fonseca, P.J. Fleming, Genetic algorithms for multiobjective optimization: formulation, discussion and generalization, in: S. Forrest (Ed.), Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, 1993, pp. 416–423.
[14] A.F. Gómez-Skarmeta, F. Jiménez, Fuzzy modeling with hybrid systems, Fuzzy Sets and Systems 104 (1999) 199–208.
[15] A.F. Gómez-Skarmeta, F. Jiménez, J. Ibáñez, Pareto-optimality in fuzzy modeling, in: Proceedings of the sixth European Congress on Intelligent Techniques and Soft Computing, EUFIT'98, Aachen, Germany, September 1998, pp. 694–700.
[16] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, 1989.
[17] J. González, I. Rojas, H. Pomares, J. Ortega, A. Prieto, A new clustering technique for function approximation, IEEE Transactions on Neural Networks 13 (1) (2002) 132–142.
[18] J.J. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.
[19] H. Ishibuchi, T. Murata, I. Türksen, Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems, Fuzzy Sets and Systems 89 (1997) 135–150.
[20] H. Ishibuchi, T. Yamamoto, Evolutionary multiobjective optimization for generating an ensemble of fuzzy rule-based classifiers, in: Proceedings of the 2003 Genetic and Evolutionary Computation Conference, GECCO'2003, 2003, pp. 1077–1088.
[21] J.S.R. Jang, ANFIS: adaptive network-based fuzzy inference system, IEEE Transactions on Systems Man and Cybernetics 23 (1993) 665–685.
[22] J.S.R. Jang, C.T. Sun, E. Mizutani, Neuro-Fuzzy and Soft Computing, Prentice-Hall, 1997, ISBN 0-13-261066-3.
[23] F. Jiménez, A.F. Gómez-Skarmeta, H. Roubos, R. Babuska, A multi-objective evolutionary algorithm for fuzzy modeling, in: Proceedings of the IX IFSA World Congress and XX NAFIPS International Conference, IFSA-NAFIPS'01, Vancouver, Canada, 2001, pp. 1222–1228.
[24] F. Jiménez, A.F. Gómez-Skarmeta, H. Roubos, R. Babuska, Accurate, transparent and compact fuzzy models for function approximation and dynamic modeling through multi-objective evolutionary optimization, in: Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science, volume 1993, Springer-Verlag, 2001, pp. 653–667.
[25] F. Jiménez, G. Sánchez, A.F. Gómez-Skarmeta, J.L. Verdegay, A multi-objective neuro-evolutionary algorithm for fuzzy modeling, in: Proceedings of the sixth Joint Conference on Information Sciences, JCIS'2002, North Caroline, USA, 2002, pp. 1222–1228.
[26] N.B. Karayiannis, G.W. Mi, Growing radial basis neural networks: merging supervised and unsupervised learning with network growth techniques, IEEE Transactions on Neural Networks 8 (6) (1997) 1492–1506.
[27] M.C. Mackey, L. Glass, Oscillation and chaos in physiological control systems, Science 197 (4300) (1977) 287–289.
[28] B. Miller, D.E. Goldberg, Genetic algorithms, selection schemes, and the varying effect of noise, Evolutionary Computation 4 (2) (1996) 113–132.
[29] J.E. Moody, C.J. Darken, Fast learning in networks of locally-tuned processing units, Neural Computation 1 (1989) 281–294.

[30] G.C. Mouzouris and J.M. Mendel, Designing fuzzy logic systems for uncertain environments using a Singular-value-QR decomposition method, in: IEEE Neural Networks Council, Proceedings of the fifth IEEE International Conference on Fuzzy Systems, New Orleans, LA, September 1996, pp. 295–301.
[31] H. Mühlenbein, D. Schlierkamp-Voosen, Predictive models for the breeder genetic algorithm: continuous parameter optimization, Evolutionary Computation 1 (1) (1993) 25–49.
[32] G. Patanè, M. Russo, The enhanced-LBG algorithm, Neural Networks 14 (9) (2001) 1219–1237.
[33] C.A. Peña, M. Sipper, Fuzzy CoCo: balancing accuracy and interpretability of fuzzy models by means of coevolution, in: J. Casillas, O. Cordón, F. Herrera, L. Magdalena (Eds.), Accuracy Improvements in Liguistic Fuzzy Modeling, Studies in Fuzziness and Soft Computing, vol. 129, Springer, 2003, pp. 119–146.
[34] J. Platt, A resource allocation network for function interpolation, Neural Computation 3 (1991) 213–225.
[35] H. Pomares, Nueva Metodología para el Diseño Automático de Sistemas Difusos (in Spanish). PhD thesis, Universidad de Granada, Spain, January 2000.
[36] H. Pomares, I. Rojas, J. González, Automatic construction of fuzzy rule-based systems: a trade-off between complexity and accuracy maintaining interpretability, in: Accuracy Improvements in Linguistic Fuzzy ModelingStudies in Fuzziness and Soft Computing, volume 129, Springer-Verlag, 2003, pp. 193–219.
[37] H. Pomares, I. Rojas, J. Ortega, J. González, A. Prieto, A systematic approach to a self-generating fuzzy rule-table for function approximation, IEEE Transactions on Systems Man and Cybernetics – Part B 30 (3) (2000) 431–447.
[38] I. Rojas, H. Pomares, J. Ortega, A. Prieto, Self-organized fuzzy system generation from training examples, IEEE Transactions on Fuzzy Systems 8 (1) (2000) 23–36.
[39] R. Rosipal, M. Koska, I. Farkaš, Prediction of chaotic time-series with a resource-allocating RBF network, Neural Processing Letters 7 (1998) 185–197.
[40] M. Srinivas, L.M. Patnaik, Adaptive probabilities of crossover and mutation in genetic algorithms, IEEE Transactions on Systems Man and Cybernetics 24 (4) (1994) 656–666.
[41] N. Srinivas, K. Dev, Multiobjective optimization using nondominated sorting in genetic algorithms, Evolutionary Computation 2 (3) (1995) 221–248.
[42] D. Thierens, D.E. Goldberg, Convergence models of genetic algorithm selection schemes, in: Y. Davidor, H.P. Schwefel, R. Manner (Eds.), Proceedings of the Third International Conference on Parallel Problem Solving from Nature, PPSN III, Lecture Notes in Computer Science, vol. 866, Springer-Verlag, Jerusalem, 1994, pp. 119–129.
[43] S. van Dijk, D. Thierens, M. de Berg, Scalability and efficiency of genetic algorithms for geometrical applications, in: M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, H.P. Schwefel (Eds.), Proceedings of the Seventh International Conference on Parallel Problem Solving from Nature, PPSN VII, Lecture Notes in Computer Science, volume 1917, Springer-Verlag, Paris, 2000, pp. 683–692.
[44] H. Wang, S. Kwong, Y. Jin, W. Wei, K.F. Man, Multi-objective hierarchical genetic algorithm for interpretable fuzzy rule-based knowledge extraction, Fuzzy Sets and Systems 149 (2005) 149–186.
[45] L.X. Wang, R. Langari, Building sugeno-type models using fuzzy discretization and orthogonal parameter estimation techniques, IEEE Transactions on Fuzzy Systems 3 (4) (1995) 454–458.
[46] L.X. Wang, J.M. Mendel, Generating fuzzy rules by learning from examples, IEEE Transactions on Systems Man and Cybernetics 22 (6) (1992) 1414–1427.
[47] B.A. Whitehead, T.D. Choate, Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction, IEEE Transactions on Neural Networks 7 (4) (1996) 869–880.
[48] J. Yen and L. Wang, An SVD-based fuzzy model reduction strategy, in: IEEE Neural Networks Council, editor, Proceedings of the 5th IEEE International Conference on Fuzzy Systems, New Orleans, LA, September 1996, pp. 835–841.
[49] J. Yen, L. Wang, Simplifying fuzzy rule-based models using orthogonal transformation methods, IEEE Transactions on Systems Man and Cybernetics – Part B 29 (1) (1999) 13–24.